

METHOD AND APPARATUS FOR CONVERTING LINEAR PREDICTIVE CODING COEFFICIENT TO REFLECTION COEFFICIENT

BACKGROUND OF THE INVENTION

5 **Field of the Invention**

This invention relates to a method and apparatus for speech processing, and more specifically to a speech coder and a method of speech coding for optimizing computing efficiency in a speech compression process.

10 **Related Art**

Speech compression, which is also known as speech coding, is a technique that reduces the amount of data to be transmitted for communication or storage. More specifically, it is a method for compressing the digital representation of speech signals, thereby reducing the number of bits that represent the speech signals. It has many practical applications in the digital transmission and digital storage of speech signals. For example, in digital transmission, speech compression is used in applications such as mobile radio, cellular telephone technology, and secure voice systems. By compressing the speech in these applications, more users can use the systems than otherwise would be possible. In the area of digital storage of speech signals, speech compression is used in applications such as digital answering machines and voice response systems. For example, in a digital answering machine, speech compression of the signals allows for the storage of longer messages than otherwise possible.

Linear predictive coding (LPC) is a method of compression that models the process of speech production. Specifically, LPC is a digital method used to predict speech samples by use of a linear combination of previous speech samples. In human speech, sound is produced in the vocal tract, which may be described as a tube having a variable diameter, or a series of tubes, each tube having a variable diameter. The LPC model is based on a mathematical approximation of this variable-diameter tube or tubes.

The algorithm for LPC involves an encoding process and a decoding process. In the encoding process, the speech coder system determines a set of parameters that model the vocal tract during the production of a given speech signal. In the decoding process, the speech coder system uses the parameters from the encoding process to build a synthesized version of the original speech signal. The decoder uses a filter to recreate the original signal, and the filter is based on a set of coefficients. These coefficients, or prediction coefficients, are taken from the original signal during the encoding process and are transmitted to the receiver for use during decoding.

During the decoding process, parameters called reflection coefficients are generated. These coefficients are derived from the LPC coefficients. A reflection coefficient represents the angle of reflection of a sound wave. When a wave arrives at a boundary between two tubes having different diameters, the wave does not simply transmit through the tube. Instead, parts of the wave are reflected from the boundary.

The angle of incidence of the sound wave on the boundary equals the angle of reflection.

In known methods and apparatuses for speech compression, a mixed excitation linear prediction ("MELP") decoder may be used to decode speech signals. The MELP uses a division operation within a recursive loop in the conversion of LPC coefficients to

reflection coefficients. Division operations, however, are difficult to process. That is, they consume much power and require many clock cycles to run. When division operations are run in recursive loops, because the loops repeat, even more computing power and clock cycles are required.

5 Therefore, it is an object of the present invention to increase computing efficiency in speech coders that perform LPC coefficients to reflection coefficients conversions and methods of doing the same.

SUMMARY OF THE INVENTION

10 In accordance with the invention, there is provided a speech coder for compressing a speech signal.

15 In an embodiment of the invention, the speech coder receives a predictor coefficient as input, stores the predictor coefficient in a first temporary storage buffer, determines a reflection coefficient from the predictor coefficient stored in the first temporary storage buffer, calculates a multiplication factor, recursively calculates a numerator and multiplies the numerator by the multiplication factor, and stops the recursive calculation after it has been performed a predetermined number of times. In this embodiment, the multiplication factor is determined outside the recursive calculation.

20 In another preferred arrangement, the speech coder calculates the numerator by use of the following equation: $\text{temp} = (b1[j] - k[i] * b1[i-j])$, where i is an integer value, j is an integer value, temp is a second temporary storage buffer for storing the numerator, $b1[j]$ is a third temporary storage buffer for storing values of the first temporary storage buffer, and $b1[i-j]$ is a fourth temporary storage buffer for storing values of the first temporary storage buffer.

In another preferred arrangement of this embodiment, the speech coder determines the reflection coefficient by amplifying the predictor coefficient stored in the first temporary storage buffer.

In yet another preferred arrangement, the speech coder calculates the multiplication factor by determining a denominator.

In still yet another preferred arrangement, the speech coder further amplifies the denominator to its largest and most accurate value.

In another preferred arrangement, the speech coder further performs a fixed-point division operation using the denominator.

In another preferred arrangement, the speech coder performs the fixed-point division operation by taking an inverse of the denominator.

In yet another preferred arrangement, the speech coder takes the inverse of the denominator by use of the following equation: $\text{Divd} = 0x4000 / n_e$, where Divd represents the multiplication factor, 0x4000 is the hexadecimal representation of 0.5, and n_e is the denominator, which represents a normalized and amplified value of a residue of the signal.

In another preferred arrangement, the speech coder further amplifies the multiplied numerator and multiplication factor.

There is also provided a method of speech coding which includes the steps of receiving a predictor coefficient, storing the predictor coefficient in a first temporary storage buffer, determining a reflection coefficient from the predictor coefficient stored in the temporary storage buffer, calculating a multiplication factor, recursively calculating a numerator and multiplying the numerator by the multiplication factor, and stopping the

recursive calculation after it has been performed a predetermined number of times. In this method, the multiplication factor is determined outside the recursive calculation.

A preferred method includes calculating the numerator by using the following equation: $\text{temp} = (b1[j] - k[i] * b1[i-j])$, where i is an integer value, j is an integer value, 5 temp is the second temporary storage buffer for storing the numerator, $b1[j]$ is a third temporary storage buffer for storing values of the first temporary storage buffer, and $b1[i-j]$ is a fourth temporary storage buffer for storing values of the first temporary storage buffer.

Another preferred method of speech coding includes the step of determining the 10 reflection coefficient by amplifying the predictor coefficient stored in the first temporary storage buffer.

Yet another preferred method includes the step of calculating the multiplication factor by determining a denominator.

Still another preferred method further includes amplifying the denominator to its 15 largest and most accurate value.

Yet another preferred method includes performing a fixed-point division operation using the denominator.

Another preferred method includes performing the fixed-point division operation by taking an inverse of the denominator.

20 Yet another preferred method includes taking the inverse of the denominator by using the following equation: $\text{Divd} = 0x4000 / n_e$, where Divd represents the multiplication factor, $0x4000$ is the hexadecimal representation of 0.5, and n_e is the

denominator, which represents a normalized and amplified value of a residue of the signal.

Another preferred method further includes amplifying the multiplied numerator and multiplication factor.

5 Also, there is provided a method of speech coding for compression of a speech signal, which includes the steps of receiving a predictor coefficient as input, storing the predictor coefficient in a first temporary storage buffer, determining a reflection coefficient from the predictor coefficient stored in the first temporary storage buffer, calculating a multiplication factor from a denominator, where the multiplication factor is defined by the equation: $\text{Divd} = 0x4000/n_e$, where Divd is the multiplication factor, 10 0.4000 is a hexadecimal representation of 0.5, and n_e is the denominator, which represents a normalized and amplified value of a residue of the signal. The method also includes the steps of recursively calculating a numerator and multiplying the numerator by the multiplication factor, where the numerator is defined by the equation: $\text{temp} =$ 15 $(b1[j] - k[i] * b1[i-j])$, where i is an integer value, j is an integer value, temp is the second temporary storage buffer, for storing the numerator, $b1[j]$ is a third temporary storage buffer, for storing values of the first temporary storage buffer, and $b1[i-j]$ is a fourth temporary storage buffer, for storing values of the first temporary storage buffer; and stopping the recursive calculation after it has been performed a predetermined number of 20 times, where the multiplication factor is determined outside the recursive calculation.

In a preferred method, the step of determining the reflection coefficient includes amplifying the predictor coefficient stored in the first temporary storage buffer.

In yet another preferred method, the method further includes amplifying the multiplied numerator and multiplication factor.

In another embodiment of the invention, the speech coder includes a predictor coefficient that is received as input, a first temporary storage buffer, where the first temporary storage buffer stores the predictor coefficient, a reflection coefficient determined from the predictor coefficient stored in the first temporary storage buffer, a multiplication factor, and a numerator, where the numerator is recursively calculated and multiplied by the multiplication factor, the recursive calculation is stopped after it has been performed a predetermined number of times, and the multiplication factor is determined outside the recursive calculation.

In another preferred arrangement, the numerator is defined by the following equation: $\text{temp} = (b1[j] - k[i] * b1[i-j])$, where i is an integer value, j is an integer value, temp is a second temporary storage buffer for storing the numerator, $b1[j]$ is a third temporary storage buffer for storing values of the first temporary storage buffer, and $b1[i-j]$ is a fourth temporary storage buffer for storing values of the first temporary storage buffer.

In another preferred arrangement of this embodiment, the reflection coefficient is determined by amplification of the predictor coefficient stored in the first temporary storage buffer.

In yet another preferred arrangement, the multiplication factor includes a denominator.

In still yet another preferred arrangement, the denominator is amplified to its largest and most accurate value.

In another preferred arrangement, the denominator is used to perform a fixed-point division operation.

In another preferred arrangement, the fixed-point division operation includes taking an inverse of the denominator.

5 In yet another preferred arrangement, the inverse of the denominator is defined by the following equation: $\text{Divd} = 0x4000 / n_e$, where Divd represents the multiplication factor, 0x4000 is the hexadecimal representation of 0.5, and n_e is the denominator, which represents a normalized and amplified value of a residue of the signal.

10 In another preferred arrangement, the multiplied numerator and multiplication factor are amplified.

Other systems, methods, features and advantages of the invention will become apparent to one skilled in the art upon examination of the following figures and detailed description. It is intended that all such additional systems, methods, features, and advantages be included within this description, be within the scope of the invention, and
15 be protected by the accompanying claims.

DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a computer system that provides the operating environment for an illustrative embodiment of the present invention.

20 Fig. 2 is a flowchart illustrating steps of a conventional method for speech compression.

Fig. 3A is a block diagram of a speech coder system that provides the operating environment for an illustrative embodiment of the present invention; Fig. 3B is a flowchart illustrating steps of a method for speech compression.

5 **DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS**

The detailed description that follows is presented largely in terms of algorithms and symbolic representations of operations on data bits and data structures within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

10 An algorithm is here, and generally, conceived to be a sequence of steps leading to a desired result. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be noted, however, that all of these and similar terms are to be associated with the appropriate physical quantities are merely convenient labels applied to these quantities.

15 Further, the manipulations performed are often referred to in terms, such as adding and comparing that are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein that form part of the present invention. The operations are machine operations. Useful machines for performing the

operations of the present invention include personal computers, hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. It should be noted that in all cases, there is a distinction between the method of operation in operating a computer and the method of computation itself. The present invention relates to method steps for operating a computer in processing electrical or other physical signals to generate other desired physical signals.

The present invention also relates to an apparatus for performing these operations. This apparatus may be specifically constructed for the required purposes, or it may be a general-purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The algorithms presented herein are not inherently related to any particular computer or other apparatus. In particular, various general purpose machines may be used with programs written in accordance with the teachings herein, or it may prove more convenient to construct a more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description given below.

Turning now to the figures, Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will first be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks

or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be practiced in distribution computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 1, an illustrative system for implementing the invention includes a conventional personal computer 102, including a processor 124, a system memory 106, and a system bus 104. The system bus 104 couples the system memory to the processor 124. The system memory 106 includes read only memory (ROM) 108 and random access memory (RAM) 110. A basic input/output system 112 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 102, such as during start-up, is stored in ROM 108. The personal computer 102 further includes a local hard disk drive 140, a magnetic disk drive 142, for reading from or writing to a removable disk, and an optical disk drive 146, for reading a CD-ROM disk 146, or for reading from or writing to other optical media. The hard disk drive 140, the magnetic disk drive 142, and the optical disk drive 146 are connected to the system bus 104 by a hard disk drive interface 130, a magnetic disk drive interface 132, and an optical disk drive interface 134, respectively. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer 102. Although the description of computer-readable media above refers to a hard disk, a removable

magnetic disk, and a CD-ROM disk, those skilled in the art will appreciate that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like may also be used in the illustrative operating environment of Fig. 1.

5 A number of program modules may be stored in the drives and RAM 110, including an operating system 114, one or more application programs 116 (e.g., word processor applications, spreadsheet applications, and presentation applications), other program modules 118 (e.g., a speech recognition engine and/or a voice recognition engine), and program data 120. The program data 120 on local hard disk drive 140 may
10 constitute speech data used in connection with the speech recognition engine and/or the voice recognition engine.

 A user of computer 102 may enter commands and information into the personal computer 102 through a keyboard 152, a microphone 166, and a pointing device, such as a mouse 150. Other input devices (not shown) may include a joystick, satellite dish,
15 scanner, or the like. These and other input devices are often connected to the processor 124 through a serial port interface 136 that is coupled to the system bus, but may be connected by other interfaces, such as a game port, a microphone input, or a universal serial bus (USB). A monitor 128 or other type of display device is also connected to the system bus 104 via an interface, such as a video adapter 126. In addition to the monitor,
20 personal computers typically include other peripheral output devices (not shown), such as speakers or printers. A speech coder 168 is also coupled to the system bus.

 The personal computer may operate in a networked environment using logical connections to one or more remote computers, such as remote computers 158 and 164.

The remote computers 158 and 164 may be servers, routers, peer devices, or other common network nodes, and the remote computers 158 and 164 typically include many or all of the elements described relative to the personal computer 102. The logical connections to computers 158 and 164 depicted in Fig. 1 include a local area network (LAN) 162 and wide area networks (WAN) 156 and 160. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, the personal computer 102 is connected to the LAN 162 through a network interface 138. When used in a WAN networking environment, the personal computer 102 typically includes a modem 154 or other means for establishing communications over the WAN. The modem 154, which may be internal or external, is connected to the system bus 104 via the serial port interface 136. A WAN connection may also be made through the network interface 138 to WAN 156. In a networked environment, program modules depicted relative to the personal computer 102, or portions thereof, may be stored in the remote memory storage device. Particularly, each remote computer 158 and 164 and their respective storage devices (not shown) can be searchable information repositories. Those of ordinary skill in the art will appreciate that the network connections shown are illustrative, and other means of establishing a communications link between the computers may be used.

Fig. 2 shows a block diagram of a conventional method for speech compression. As shown in Fig. 2, the method 200 begins at start step 205 and proceeds to step 210. In step 210, the variable *i* is initialized. That is to say, a value of 1 is assigned to the variable *i*.

Next, in step 215, an evaluation of the condition $i \leq p$ occurs. The method 200 determines whether i is less than or equal to p , where p represents the predictor order. The value of p may range from 0 to an infinite integer value. For this implementation, a 10 order value is chosen. If i is less than or equal to p , then the method 200 proceeds to step 220 and executes the statement that assigns the linear predictor coefficient, $a[i]$, to temporary storage buffer $b[i]$. After the execution statement in step 220, i is incremented in step 225, and the method 200 loops back to step 215 and checks the condition again. This loop continues until the condition evaluates to false, at which point the method 200 proceeds to step 230, where the variable i is re-initialized. That is to say, the value of p , or 10 in this example, is assigned to i .

Next, the method 200 proceeds to determine the reflection coefficients. First, in step 235, the evaluation of the condition $i > 0$ occurs. The method 200 determines whether i is greater than 0, and if i is greater, then the method 200 proceeds to step 240. In step 240, mathematical manipulations are conducted to calculate the energy of a residual signal. The manipulations include amplifying the value stored in the temporary storage buffer $b[i]$, which contains the value of the predictor coefficient $a[i]$, so as to obtain the reflection coefficient $k[i]$. The reflection coefficient $k[i]$ is then used to calculate the residual signal e . In addition, the variable ONE_Q15, which is approximately equal to 1 in Q15 format representation, is used in the calculation. The energy of the residual signal e is calculated from the variable e . After the energy is calculated, the method 200 proceeds to step 245. There, the variable j is initialized to a value of 1.

The method 200 then proceeds to step 250, and the evaluation of the condition $j < i$ occurs. If j is less than i , then in step 255, the value in the temporary storage buffer $b[j]$ is assigned to the temporary storage buffer $b1[j]$. Next, in step 260, the variable j is incremented by 1. After the execution of step 260, the method 200 loops back to step 250 and checks the condition again. This loop continues until the condition evaluates to false, at which point the method 200 proceeds to step 265, where the variable j is re-initialized to a value of 1.

After the execution of step 265, the method 200 proceeds to step 270. There, the evaluation of the condition $j < i$ occurs. If j is less than i , then in step 275 through step 280, mathematical manipulations are conducted to solve the equation:

$$(1) \quad b[j] = (b1[j] - k[i] * b1[i-j]) / n_e.$$

First, in step 275, the numerator $(b1[j] - k[i] * b1[i-j])$ is calculated. Before the fixed-point division operation, the method 200 must verify that both the numerator and the denominator are positive values and that the numerator is less than or equal to the denominator. As such, the method 200 takes the absolute value of the numerator to get a positive value and normalizes the denominator to get a positive value. A shift left operation is performed on the denominator, thereby amplifying it to its largest accuracy. The denominator must be greater than 4000H.

Next, in step 277, the method 200 evaluates the condition $n_temp > n_e$. If the numerator is greater than the denominator, then the method 200 proceeds to step 278 to perform a shift right operation on the numerator, which effectively compresses the numerator. This operation ensures that the numerator is less than or equal to the denominator. Also, the value of the variable shift is incremented in step 278. The

method 200 then proceeds to step 280, which performs the fixed-point division operation of equation (1). If the numerator is not greater than the denominator, then the method 200 proceeds directly to step 280.

As shown in step 280, a division operation, which divides $(b1[j] - k[i] * b1[i-j])$ by n_e is performed. This division operation occurs within the recursive loop prompted by condition step 270. After the execution of step 280, the method 200 proceeds to step 285, where the temporary storage buffer temp1, which represents the numerator $(b1[j] - k[i] * b1[i-j])$, divided by the denominator n_e , is amplified. The result is stored in the temporary storage buffer $b[j]$.

The numerator must be returned to a negative value if it started as a negative value. Therefore, in step 287, method 200 evaluates the condition $temp^e < 0$. If the condition evaluates to true, then $b[j]$ is negated. The method 200 then increments j by 1 in step 290, loops back to step 270, and checks the condition again. This loop continues until the condition evaluates to false, at which point the method 200 proceeds to step 295, where the variable i is decreased by 1.

After step 295, the method 200 loops back to step 235 and checks the condition $i > 0$ again. This loop continues until the condition evaluates to false, at which point the method 200 ends at step 298.

The method 200 includes steps, which may be implemented in C programming language. An exemplary program that performs the steps of the conventional method 200 is provided after the specification and before the claims (*see* Part I of the Computer Program Listings).

Fig. 3A is a block diagram of a speech coder system that provides the operating environment for an illustrative embodiment of the present invention. As shown in Fig. 3A, the speech coder system 302 includes an encoder 304 and a decoder 306. A speech signal is input into the encoder 304, and the encoder 304 determines parameters that model the speech signal. The predictor coefficients are a set of parameters that are output from the encoder 304. The decoder 306 receives the predictor coefficients as input, and uses the predictor coefficients to determine the reflection coefficients. The decoder 306 uses a filter to recreate the original signal, and the filter is based on the coefficients. The decoder 306 then outputs the synthesized speech.

Fig. 3B shows one method of implementing the present invention. As shown in Fig. 3B, the method 300 begins at start step 305 and proceeds to step 310. In step 310, the variable i is initialized. That is to say, a value of 1 is assigned to the variable i .

Next, in step 315, an evaluation of the condition $i \leq p$ occurs. The method 300 determines whether i is less than or equal to p , where p is the predictor order. For this implementation, a value of 10 has been chosen, but the value may be any integer from 0 to an infinite integer value. If i is less than or equal to 10, then the method 300 proceeds to step 320 and executes the statement that assigns the predictor coefficients, $a[i]$, to temporary storage buffer $b[i]$. After the execution statement in step 320, i is incremented in step 325, and the method 300 loops back to step 315 and checks the condition again.

This loop continues until the condition evaluates to false, at which point the method 300 proceeds to step 330, where the variable i is re-initialized, this time, to a value of p , which may be 10 in this implementation. Again the value of p may be replaced with another integer value.

Next, the method 300 proceeds to step 335, and the evaluation of the condition $i > 0$ occurs. The method 300 determines whether i is greater than 0, and if i is greater, then the method 300 proceeds to step 340. In step 340, mathematical manipulations are conducted to calculate the energy of a residual signal. The manipulations include

5 amplifying the value in the temporary storage buffer $b[i]$, which contains the value of the predictor coefficient $a[i]$, so as to obtain the reflection coefficient $k[i]$. The reflection coefficient $k[i]$ is then used to calculate the energy of the residual signal e . In addition, the variable ONE_Q15, which is approximately equal to 1 in Q15 format, is used in the calculation. The energy of the residual signal e is calculated from the variable e . After

10 the energy is calculated, the method 300 proceeds to step 345. There, the variable j is initialized to a value of 1. The method 300 then proceeds to step 350, and the evaluation of the condition $j < i$ occurs. If j is less than i , then in step 355, the value in the temporary storage buffer $b[j]$ is assigned to the temporary storage buffer $b1[j]$. Next, in

15 step 360, the variable j is incremented by 1. After the execution of step 360, the method 300 loops back to step 350 and checks the condition again. This loop continues until the condition evaluates to false, at which point the method 300 proceeds to step 365 where the signal represented by the variable e is normalized and then amplified. The variable e is normalized because the denominator must be a positive value, and the denominator must be amplified to the largest accuracy. That is, the denominator must be greater than

20 4000H. In the next step 370, the method 300 calculates a multiplication factor, Divd. As shown in Fig. 3B, the multiplication factor is calculated using the following equation:

$$(2) \quad \text{Divd} = 0x4000 / n_e.$$

The 0x4000 value is a predetermined value that represents the hexadecimal representation for the decimal number 0.5. This factor is used to satisfy the requirements for a fixed-point division operation, such that the denominator, n_e , must be amplified to its largest most accurate value. After the calculation of the multiplication factor, the method 300 proceeds to step 375 where the variable j is initialized to a value of 1.

After the execution of step 375, the method 300 proceeds to step 380. There, the evaluation of the condition $j < i$ occurs. If j is less than i , then in step 385, mathematical manipulations are conducted to solve the equation:

$$(3) \quad \text{temp} = (b1[j] - k[i] * b1[i-j]) * \text{Divd.}$$

This equation (3) calculates the temporary storage buffer temp , which stores the product of the numerator and the multiplication factor. In the numerator $(b1[j] - k[i] * b1[i-j])$, i is an integer value, j is an integer value, $b1[j]$ is a temporary storage buffer for storing values of the temporary storage buffer that stores the predictor coefficient, and $b1[i-j]$ is a temporary storage buffer for storing values of the temporary storage that stores the predictor coefficient. As shown in step 385, this step does not include a division operation. That is to say, in the present invention, the division operation, which divides the numerator $(b1[j] - k[i] * b1[i-j])$ by the denominator n_e is not performed. The inverse of n_e is calculated outside the recursive loop prompted by condition step 380. Specifically, the inverse of n_e is calculated outside the recursive loop by equation (2), which calculates Divd. In step 385, the numerator $(b1[j] - k[i] * b1[i-j])$ is calculated and stored in the temporary storage buffer temp . Then the numerator is multiplied by the inverse of the denominator, which is the multiplication factor Divd. A shift left operation is performed on the product, thereby multiplying the product by 2, so as to compensate

for the 0.5 multiplication factor used in the calculation of Divd. After the execution of step 385, the method 300 proceeds to step 390, where j is incremented by 1. The method 300 loops back to step 380 and checks the condition again. This loop continues until the condition evaluates to false, at which point the method 300 proceeds to step 395, where the variable i is decreased by 1.

After step 395, the method 300 loops back to step 335 and checks the condition $i > 0$ again. This loop continues until the condition evaluates to false, at which point the method 300 ends at step 398.

The present invention removes a division operation from a recursive calculation within the conversion process in a mixed excitation linear prediction decoder. As a result, less power is consumed and fewer clock cycles are required to run.

The method 300 includes steps, which may be implemented in C programming language. An exemplary program that performs steps that implement the method 300 is provided after the specification and before the claims (*see* Part II of the Computer Program Listings).

While various embodiments of the present invention have been described, it will be apparent to those of ordinary skill in the art that many more embodiments and implementations are possible that are within the scope of this invention. Accordingly, the invention is not to be restricted except in light of the attached claims and their equivalents.

COMPUTER PROGRAM LISTINGS

I. EXEMPLARY PROGRAM OF THE PRIOR ART

```
/*  
5 |  
| NAME: lpc_pred2refl_bak() ( original code )  
|  
| get reflection coefficients from the predictor coefficients  
| Input:  
10 | a- the predictor coefficients Q12  
| p- the predictor order Q15  
| Output:  
| k- the reflection coefficients  
| Returns:  
15 | energy - energy of residual signal Q15  
|  
*/  
Shortword lpc_pred2refl_bak(Shortword *a, Shortword *k, Shortword p)  
20 {  
| Shortword b[11], b1[11], e;  
| Shortword energy = ONE_Q15;  
| Shortword i, j;  
| Shortword shift;  
25 | Shortword temp, temp1, n_temp, n_e;  
|  
| //==== equate temporary variables (b = a) ====  
| for(i=1; i <= p; i++)  
30 | {  
| | b[i] = a[i];  
| | }  
| //==== compute reflection coefficients ====  
| for(i=p; i > 0; i--)  
35 | {  
| | k[i] = shl(b[i],3); // Q12 -> Q15  
| | e = sub(ONE_Q15,mult(k[i],k[i]));  
| | energy = mult(energy,e);  
| | for(j=1; j < i; j++)  
40 | | {  
| | | b1[j] = b[j];  
| | | }  
| | }
```

```
//==== b[j] = (b1[j] - k[i]*b1[i-j])/e ====
```

```
5 for(j=1; j < i; j++)
```

```
{  
  //==== b[j] = (b1[j] - k[i]*b1[i-j]) ====
```

```
    temp = mult(k[i],b1[i-j]);
```

```
    temp = sub(b1[j],temp);    //calculate numerator
```

```
10 //==== check signs of temp and e before division ====
```

```
    //(before fix-point division operation, must verify that both the numerator and  
    // denominator are positive values, and numerator <= denominator)
```

```
15 n_temp = abs_s(temp);    // take absolute value of  
                                // numerator to get a  
                                // positive value
```

```
20 shift = norm_s(e);        // normalize denominator  
                                // (denominator must be a  
                                // positive value)
```

```
25 n_e = shl(e, shift);      // amplify denominator to largest accuracy;  
                                // denominator must be > 4000H
```

```
    if (n_temp > n_e)        // verify that numerator > denominator  
    {  
30      n_temp = shr(n_temp,1);  
      shift = add(shift,1);  
    }
```

```
    //calculate fixed-point division operation n_temp/n_e
```

```
35 temp1 = divide_s(n_temp,n_e);
```

```
    b[j] = shl(temp1,shift);    // b[j] in Q12
```

```
40    // return numerator to negative value if it started as a negative value
```

```
    if ((temp^e)<0) b[j] = negate(b[j]);
```

```
    }
```

```
    }  
    MEM_FREE(FREE,b1);  
5    MEM_FREE(FREE,b);  
    return(energy);  
} /* LPC_PRED2REFL_BAK */
```

10

II. EXEMPLARY PROGRAM OF THE PRESENT INVENTION

```
/*  
NAME: lpc_pred2refl_k() ( simplified code K; error rate 1 bit )  
  
get reflection coefficients from the predictor coefficients  
Input:  
a- the predictor coefficients Q12  
p- the predictor order      Q15  
Output:  
k- the reflection coefficients  
Returns:  
energy - energy of residual  Q15  
*/
```

```
Shortword lpc_pred2refl_k (Shortword *a, Shortword *k)
```

```
{  
    Shortword b[11], b1[11];  
    Shortword energy = ONE_Q15;  
    Shortword i, j;  
    Shortword shift;  
    Shortword temp, temp1, n_temp, n_e;
```

```
    //==== equate temporary variables (b = a) ====  
    for(i=1; i <= p; i++)  
    {
```

```
        b[i] = a[i];
```

```
    }
```

```
//==== compute reflection coefficients =====
```

```
for(i=p; i > 0; i--)
```

```
{
```

```
    k[i] = shl(b[i],3);
```

```
    e = sub(ONE_Q15, mult(k[i], k[i]));    // e
```

```
    energy = mult(energy,e);    //energy=1-k[i]^2
```

```
        for (j=1; j<1; j++)
```

```
        {
```

```
            b1[j] = b[j];
```

```
        }
```

```
    shift = norm_s(n_e); // denominator must be a positive value
```

```
    n_e = shl(n_e, shift); // amplify denominator to the largest accuracy;  
                          // denominator must be > 4000H
```

```
    divd = divide_s((short)0x4000,n_e); // inverse of denominator  
                                         // 0.5/n_e
```

```
//==== b[j]=(b1[j]-k[i]*b1[i-j]) * (1/e) =====
```

```
for(j=1;j<i;j++)
```

```
{
```

```
//==== (b1[j] - k[i] * b1[i-j]) =====
```

```
    temp = mult(k[i], b1[i-j]);
```

```
    temp = sub(b1[j], temp); // calculate numerator
```

```
//==== (numerator) * (inverse of denominator) * 2
```

```
    temp1 = mult_sft(temp, divd, 1);
```


b[j] = shl(temp1, shift); // b[j] in Q12

```
    }  
5    return(energy);  
} /* LPC_PRED2REFL_K */
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216